SPECIFICATION

Electronic Version 1.2.8

Stylesheet Version 1.0

Method For Detecting Current Client-Side Browser Encoding

Background of Invention

[0001] The world wide web is being used by millions of users around the world, with different languages. TCP/IP and HTTP protocols transmit data between server and client, in most cases not having the exact knowledge of the language and encoding that the client-side user uses. While Unicode covers all known languages and characters, its encodings, UTF-8 and UTF-16, are very rarely used as a standard for information exchange. Instead, some languages use several different encodings. For instance, there are two widely-used Russian encodings, and two more, less widely used. Many languages have one encoding for Windows operating system and another for DOS. Linux and Unix often use one more encoding; e.g. in Japanese, Shift JIS is widely (but not always) used on Windows, and EUC-JP is widely (but not always) used on Linux and Unix.

[0002] Ordinary users around the world do not know and often do not care what encoding they have. It can be a problem when the user downloads a page in a different encoding, but this is solved by specifying page encoding inside HTML. When the users sends a form to the server, though, the server cannot find out the client-side encoding, and can either guess, or keep the data as received, in whatever encoding it was.

[0003] This makes searches in international databases almost impossible: for instance, the same set of codes can correspond to different characters in different languages. This also makes it impossible to store data in the server databases in encoding-independent way (which basically means in Unicode).

[0004] Some web sites solve this problem by having different pages for different languages; which is still a partial solution for the languages that have several encodings; and since the users, as experience shows, do not know their encoding, the data they supply cannot be always correctly parsed.

[0005] Another solution is to retrieve from HTTP request header encodings that are enabled on the client side. This gives only a hint on which languages can be installed on the user's computer. In some occasions it can be enough, when there is one language that has one encoding; in other occasions it is not enough, for instance in the case of a computer being used for Japanese-Ukrainian translation. In this latter case the computer will have at least two languages installed, each of the languages having three different encodings: we have to choose between 7 (add English) encodings.

[0006] If the browsers made current encoding available in a JavaScript object on the web page, or to the server in the HTTP request, this would be a solution, but unfortunately this is not so: browsers do not provide this information.

[0007]

[t1]

Related US Patents:

Patent Number	Date	Author
5944790	July, 1996	Levy

[0008]

[t2]

Other References

Peter Kent, John Kent	"Official Netscape JavaScript 1.2 Book, Second Edition", Ventana, 1997.
The Unicode Consortium, Joan Aliprand, Julie Allen, Rick McGowan, Joe Becker, Michael Everson, Mike Ksar, Lisa Moore, Michel Suignard, Ken Whistler, Mark Davis, Asmus Freytag, John Jenkins	"The Unicode Standard, Version 3.0", The Unicode Consortium.
Nadine Kano	"Developing International Software for Windows 95 and Windows NT", Microsoft Press, 1995

Summary of Invention

[0009] The present invention solves the problem of browser encoding detection. The result of detection can be used in a JavaScript program or in a Java applet to adapt the contents depending on the encoding. The result can also be passed to the server, either in consequent HTTP requests, or with the form data. If the form data are accompanied by the encoding name, then the data can be uniquely converted into encoding-neutral Unicode strings.

[0010] The method consists of creating an invisible form in the HTML document, with the only hidden input field that contains Unicode character codes for a sample Unicode string, and matching parts of the sample Unicode string with characters or sequence of characters in various specific encodings; when the characters match, the encoding is detected.

Detailed Description

[0011] The browser encoding is detected in a piece of JavaScript code that is placed in the very top of HEAD part HTML page, before any body text is written to the document. First, a form is written to the document, with th hidden input the value of which is the sample Unicode string, e.g.:document.write("<form name=VP_encoding><input name=t type=hidden value='АÀĄĎ΅ğ -個 '></input></form>"); JavaScript also contains a function, VP_getEncoding(), that returns the current encoding name.

[0012] The function works like this:First, it splits the sample Unicode string into two samples, one for multi-byte encodings (*multi-byte sample*), another for Utf-8 and single-byte encodings (*single-byte sample*).

[0013] The second step detects Utf-8 encoding by comparing the single-byte sample to the same string directly encoded using Utf-8. If the comparison is positive, the algorithm stops.

[0014] The third step compares the multi-byte sample string to the same string encoded in Big5 Chinese, GBK Chinese, EUC_TW Chinese, EUC_JP Japanese, SJIS Japanese (the list can be easily extended). Note that the multi-byte sample string is padded with space character, to make it a valid sequence of bytes when the encoding is Utf-8.

[0015] The fourth step compares one or two characters of single-byte sample strings to the characters directly encoded using different single-byte encodings. Note that the character cannot be stored alone in the string, but instead has to be padded with space character, to make the sequence legal in Utf-8 encoding. The set of encoding samples can be easily expanded.

[0016] If the fourth step does not detect the encoding, "?"is returned.

[0017] The function VP_getEncoding(), can be later used in JavaScript later on the page, or in event handling routines, and the result can be passed back to the server if needed.

Program Listing Deposit

```
<HTML><HEAD><TITLE>Encoding test</title><META HTTP-EQUIV="Pragma" CONTENT="no-cache"

<%
  int []det1b = new int[] { 1040, 192, 260, 270, 901, 287 };
//                          Cyr  West  CtrE Balt GR   Turk
//                                     (with prev)
  int []det2b = new int[] { 0x500b };
//                         dbl/utf
%>

<form name="_unicode_">
<input name="t1b" type="hidden"  value="<%
  for (int i = 0; i < det1b.length; i++) {
    out.print("&#" + det1b[i] + ";");
  }
%>"></input>
<input name="t2b" type="hidden"  value="<%
  for (int i = 0; i < det2b.length; i++) {
    out.print("&#" + det2b[i] + ";");
  }
%> "></input>
</form>
<hr>

<script language="javascript">

<%  String[] b2 = new String[] {"UTF8",    "\u00e5\u0080\u008b",
                                "Big5",    "\u00ad\u00d3",
```

```
                                "GBK",     "\u0082\u0080",
                                "EUC_TW",  "\u00d4\u00b6",
                                "EUC_JP",  "\u00b8\u00c4",
                                "SJIS",    "\u008c\u00c2" };

    String[] b1 = new String[] {
                                "UTF-8",                        "\u00d0\u0090\u00c3\u008
                                "Central-European Windows",  "  \u00a5\u00cf ",
                                "Central-European ISO",      "  \u00a1\u00cf ",
                                "Baltic ISO",                "  \u00a1 ",
                                "Cyrillic DOS",              "\u0080 ",
                                "Baltic Windows",            "  \u00c0 ",
                                "Cyrillic Windows",          "\u00c0 ",
                                "Cyrillic KOI-8",            "\u00e1 ",
                                "Cyrillic ISO",              "\u00b0 ",
                                "Turkish",                   " \u00c0   \u00f0  ",
                                "ISO_8859_1",                " \u00c0 ",
                                "Greek ISO",                 "    \u00b5 ",
                                "Greek Windows",             "    \u00a1 ",
                               };

%>

  function VP_getEncoding() {
    var encoding = "?";
    var t1 = document.forms._unicode_.t1b.value;
    var t2 = document.forms._unicode_.t2b.value;
<%  // Check for multibyte stuff
    for (int i = 0; i < b2.length; i+=2) { %>
      <%= i > 0 ? "else " : "" %> if (t2 == "<%= b2[i+1] %> ") {
        encoding = "<%= b2[i] %>";
      }<%
    }
```

```
  // Check for single-byte stuff
    for (int i = 0; i < b1.length; i+=2) { %>
      if (encoding == "?") {
<%
        String originalSample = b1[i+1];
        String workingSample  = "";
        int[] chosen = new int[originalSample.length()];

        for (int j = 0; j < originalSample.length(); j++) {
          char c = originalSample.charAt(j);
          if (c != ' ') {
            chosen[workingSample.length()] = j;
            workingSample += c;
          }
        }

        if (workingSample.length() == originalSample.length()) {
%>
        if (t1 == "<%= originalSample %>") {
<%
        } else {
%>
          test = "<%= originalSample %> ";
          if ( <%
          for (int j = 0; j < workingSample.length(); j++) {
%><%= j > 0 ? ") && " : ""%> (t1.charAt(<%= chosen[j] %>) == test.charAt(<%= chosen[
          }%>)) {
<%      } %>
        encoding = "<%= b1[i]%>";
        }
      }
  <%}%>
    return encoding;
  }
```

```
document.write("Encoding is <font color=red><b>" + VP_getEncoding() + "</b></font><b
</script>
</BODY>
</HTML>
```